Coevolution of heteregoneous multi-robot teams for temporally coupled tasks

Paper 578

ABSTRACT

Autonomous multi-robot teams offer key advantages over single robots in many exploration tasks. They increase coverage and explore larger areas, spread risk via graceful degradation with robot failures, and enable robot specialization. However, coordinating multiple robots is a challenging problem, particularly when tasks require multiple types of robots (heterogeneity) and a strict ordering of the joint action (tight temporal coupling). Many environmental monitoring task fall under this category, where a smaller, cheaper, and more agile Unmanned Aerial Vehicle (UAV) explores the environment before a heavier, more sophisticated robot travels to the location and tags an endangered animal. In this problem, it is imperative that once an animal is scouted, the second robot reach the location and tag the animal within a specified time window, before the animal is likely to move. In this paper, we utilize a cooperative coevolutionary algorithm modified to evolve Neuroevolution of Augmenting Topologies (NEAT) neural networks as robot policies, in conjunction with difference evaluation function to evolve coordination policies for such temporally coupled heterogeneous domains. Our results demonstrate that policies evolved using the difference evaluation successfully capture and account for the strictly ordered temporal coupling in the heterogeneous joint action space, and result in up to 20% improvement in system performance, when compared to policies evolved with global evaluation.

CCS Concepts

•**Computing methodologies** → **Multi-agent systems;**

Keywords

Difference Rewards, Coevolutionary Algorithms, Robotics, NEAT, Multi-robot teams

1. INTRODUCTION

Autonomous multiagent teams can accomplish complex tasks in highly dynamic and stochastic environments improving on both speed and effectiveness over single robot systems. However, multiagent coordination is a complex control problem especially in heterogeneous teams with different capabilities. The difficulty of the task is further exacerbated when the objective is temporally sensitive and requires a strictly ordered joint action within the heterogeneous team of agents. In such systems, the performance of the multiagent team is highly sensitive to the level of coordination captured and facilitated by the agent's individual policies.

In many multiagent problems such as space exploration, environmental monitoring, or underwater exploration, only a high level representation of the mission is available, and the agents often only have access to local sensor information to enact their policies. The locations of Points of Interest (POIs) are unknown *a priori* and are often dynamic. These circumstances often preclude for centralized control that can decompose the overall task component by component, and provide explicit instruction to each agent.

Further, most tasks often require, or benefit from a heterogeneous team of agents with varying capabilities. A space exploration task [6] can benefit from having a simple, light and agile robot that can scout for POIs before a heavier and less mobile, but more capable robot travels to the location and collect samples or runs experiments. Such heterogeneous tasks further complicate centralized and deterministic coordination, often leading to distributed policy learning being the preeminent choice of forming coordination policies.

A similar decomposition of tasks facilitated through varying capabilities and limitations is often utilized in environmental monitoring tasks where the POIs are endangered wildlife that might need tagging or medical interventions by ground resources, often traveling at lower visibility conditions, or through rough terrain. Unlike the space exploration task however, where the environment is mostly static, the environmental monitoring domain is volatile and imposes temporal constraints which require a strictly ordered series of joint action. It is imperative that once an animal is scouted, the second robot travels to the location and tags the animal within a specified time window, before the animal is likely to move. In a volatile environment like this, a strict temporal coupling exists between the two heterogeneous agents' actions such that a correct joint action that maximizes system utility is also conditioned on meeting the strict temporal constraints.

Distributed policy learning is often utilized to form coordination policies in multiagent domains and has been demonstrated to produce effective team performance for spatially coupled homogeneous tasks [3][11][17]. Reward shaping techniques [7] in particular have been successful in finding good

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

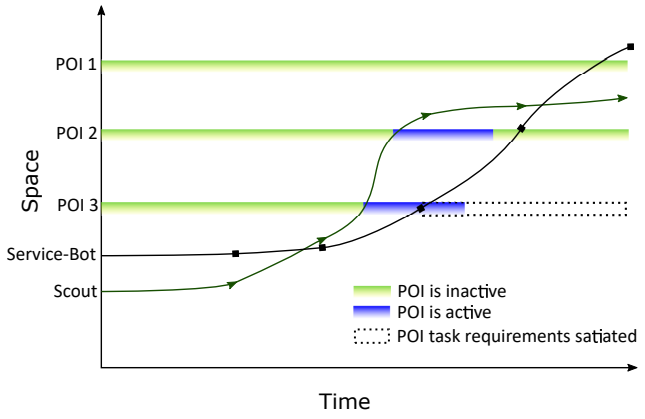


Figure 1: Space-Time illustration of the temporally constrained heterogeneous coordination problem. POI 1 is never activated by the Scout, thus the Service-bot's arrival to it is futile. POI 2 is triggered but the Service-bot arrives after the trigger time has passed thus the POI's observation requirement remains unsatisfied. POI 3 is activated by the Scout and is promptly accessed by the Service-bot within the trigger time, which satisfies its observation requirements

solutions for implicit multiagent coordination problems. Difference evaluation [2] is one such reward shaping technique that utilizes counterfactuals to isolate an individual's marginal contribution to the team. Difference evaluation effectively filters out the noise and provides a clear learning signal which is both aligned to the team's utility and sensitive to the individual's own actions. It has been demonstrated to be an effective technique in many multiagent coordination tasks [2][6][5] including a loosely coupled heterogeneous rover domain [10].

In this paper, we extend the application of the difference evaluation's credit resolving capabilities to demonstrate heterogeneous multiagent coordination on a tight temporally coupled task. We employ a heterogeneous multi-robot coordination domain with tight temporal coupling requirements, as a structured proxy for an environmental monitoring task. Apart from requiring joint action from a heterogeneous team of robots, our domain enacts temporal coupling by imposing a time constraint between the joint action required in strict order to successfully satisfy a POIs observation requirement. Following activation by our first class of robot (Scout), the second class of robot (Service-bot) needs to access the POI within a certain observability window, in order to successfully satisfy the POI's observation requirement. Failing any such event, the POI transitions back into an inactive state, and the entire joint action has to be repeated. This is illustrated in Figure 1. This formulation ensures that tight coordination between the heterogeneous robots in terms of both space and time is crucial to solve the task.

The key contribution of this work is to utilize a cooperative coevolutionary algorithm (CCEA) [8] modified to evolve NeuroEvolution of Augmenting Topologies (NEAT) [15] neural networks, in conjunction with difference evaluation to evolve coordination policies for the heterogeneous robot domain with tight temporal coupling. NEAT's ability to develop complex yet diverse sets of control policies for each robot operating under the parallel architecture of a cooperative coevolutionary algorithm with leniency gives our approach a complex, yet flexible and fast platform for evolving individual robot policies. This platform when combined with the difference evaluation technique, which is able to successfully capture and account for the strictly ordered temporal coupling in heterogeneous joint action space, leads to effective coordination policies for these difficult domains.

The rest of this paper is organized as follows. Section 2 provides the background on difference evaluation functions, CCEA, and NEAT. Section 3 describes our heterogeneous multi-robot coordination domain with tight temporal coupling requirements and Section 4 explains our training algorithm. Section 5 describes a number of experiments which probe the performance of our approach and Section 6 concludes by discussing the strengths and weakness of our approach, and highlighting future areas of research including a pathway to hardware implementation.

2. RELATED WORK AND BACKGROUND

Extending single robot approaches to multi-robot coordination tasks presents challenges in ensuring that each robot learns policies that are beneficial to the entire team. A viable decomposition of the team goal is rarely available or easily computable, which necessitates autonomous agents that can learn, on the fly. Stone et al.[16] highlights this need for collaboration of autonomous agents without preordianed protocols for coordination. Distributed policy learning is well suited to these kinds of problems and has received much attention in the field of multi-robot coordination. Colby et. al [6] utilized a cooperative coevolutionary algorithm to train multi-robot teams exploring an unknown space environment with humans in the loop. Yong and Miikkulainen [19] utilized Multi-Agent Enforced Subpopulations to evolve a team of predators in a prey capture task. Ishiwaka et al. [9] utilized Q-learning in a similar prey-capture task with heterogeneous team of predators. Knudson and Tumer [10] coevolved multi-robot teams to solve a tightly coupled task requiring multiple robot observations and extended it to heterogeneous multi-robot teams with varying capabilities.

While most of these work focus on homogeneous multi-robot coordination, some extend their focus to heterogeneous teams where a joint action from agents with varying capabilities is required to complete the task. However, temporal coupling, which constrains the required joint action temporally, has received scant attention. In this paper we focus our investigation on this facet of task complexity, by employing a heterogeneous multi-robot coordination domain with tight temporal coupling requirements. The following sections provide detail on Cooperative Coevolutionary Algorithms (CCEA), Neuroevolution of Augmenting Topologies (NEAT) and Difference Evaluation functions.

2.1 Cooperative Coevolutionary Algorithms

Cooperative Coevolutionary Algorithms (CCEAs) are an extension of Evolutionary Algorithms (EA) that deal with multiple evolving sub-populations, and have been shown to perform well in cooperative multiagent domains [8]. Multiple populations evolve in parallel, with each population developing a policy for one of the robots in the team. Policies from each population are drawn to form a team of robots, and the overall performance of this team is evaluated using a fitness function $F(z)$, where z is the joint state of all robots in the team. This fitness is then assigned to each policy in the team. The key difference between a CCEA and

an EA is the assignment of fitness to each evolving agent. In a traditional EA, the fitness of a policy is simply the system performance attained by that policy. However, in a CCEA, all the agents in the team affect the overall system performance; this means that the fitness of an agent's policy is based on its interactions with all other agents it collaborates with, resulting in context-dependent and subjective fitness assignment [5].

To alleviate this problem, Panait et al. introduced the concept of lenient learners [12] in coevolutionary algorithms, which are agents that forgive possible mismatched teammate actions that result in poor team performance. Lenient learners are shown to receive more accurate feedback on their individual policies which increases the likelihood that the learning process converges to an optimal solution rather than a robust one. Leniency is implemented by pairing a policy with multiple set of collaborators and assigning the highest fitness achieved across the many runs, to the policy at the end of each epoch.

2.2 NeuroEvolution of Augmenting Topologies (NEAT)

NeuroEvolution of Augmenting Topologies often referred to as NEAT [15][14] is a computational framework for evolving neural networks by altering the network topology as well as the neural weights. Each Artificial Neural Network (ANN) is stored as a genome, whose connection weights, nodes and activation functions can be modified after each generation. The genome translates into an ANN which is tested for the optimization task returning an associated fitness value. Following fitness assignment for its population of genomes, selection of the next generation of genomes takes place with the probability of selection for each genome directly proportional to its relative fitness value in the population. NEAT also keeps track of the historical markings associated with each member throughout successive generations often referred to as the innovation number. This allows NEAT to perform crossover amongst a diverse set of network topologies present in the population without cost-prohibitive computations. NEAT also utilizes this feature to separate the population into species, localizing competition and preserving topological diversity throughout generations. A key component of NEAT is its progressive search towards complexity; it starts with the simplest, most minimal network structures then gradually adds complexity by creating and expanding new nodes and links. This allows for an efficient trajectory for search through a enormous space. A sum of these features combine to make NEAT a fast, memory efficient and versatile algorithm for evolutionary computation which has successfully been applied to many complex tasks [1][13][18].

2.3 Difference Evaluation

Difference evaluation is a reward shaping technique that gauges an individual's marginal contribution to the team. For an agent i, the difference evaluation is computed as in [2] shown here in Equation 1:

$$D(i) = G(z) - G(z_{-i} \cup c_i) \tag{1}$$

where $G(z)$ is the global system evaluation of the full team, z_{-i} are the states and actions that were unaffected by agent i and c_i is the counterfactual term which is a fixed vector that replaces agent i. Intuitively, the second term can be though of as the global system evaluation if agent i were to be replaced by a counterfactual agent. The difference evaluation, then gives agent i's contribution to the global system evaluation function. In this equation, note that:

$$\frac{\partial D_i(z)}{\partial a_i} = \frac{\partial G(z)}{\partial a_i} \tag{2}$$

where a_i represents the action of agent i. The equality shown in Equation 2 gurantees that an agent acting to increase its difference evaluation function will also simultaneously increase the overall global system performance by the virtue of the same action. This property is termed alignment and is a critical component of the shaped reward. Additionally, in Equation 1 the second term eliminates the portions of the global system performance that are unrelated to agent i's actions, resulting in an improved signal to noise ratio. It effectively helps filter out the effect of other agent's action in the computation of the global system performance and provides a better mapping of agent i's own actions to overall performance. This property is termed sensitivity and is highly beneficial in learning good individual policies for an agent. The two properties, namely alignment and sensitivity, guranteed by difference evaluation functions has previously been demonstrated to lead to good agent specific rewards [2] and result in superior learning performance.

3. TEMPORALLY COUPLED MULTI-ROBOT COORDINATION

The domain used in this paper is a variant of the rover domain used in [10]. A detailed explanation of our domain, including system dynamics and the global team objectives used is included in this section.

3.1 Domain

In our domain, a collection of heterogeneous robots on a two dimensional plane aim to observe POIs scattered across the domain. Figure 2 illustrates our domain between a couple of timesteps. Prior knowledge of POIs, such as their location is not known and robots must learn to coordinate such that the heterogeneous team's utility function is maximized. The team's combined utility function is often referred to as the system reward and is simply defined here as the percentage of POIs at the end of an episode, which have satisfied their observation requirements. The functional form is shown in Equation 3.

$$G(N) = \frac{1}{N}\sum_{i=1}^{N} V(i)\cdot I(i) \; where \; I(i) = \begin{cases} 1 & \text{if } POI_i \text{ is observed} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Here N represents the total number of POIs in the system and I is an indicator function which counts the total number of POI's that were successfully observed. $V(i)$ is the value of POI i, set to 1 for all POIs here.

The domain is implemented to be discrete with deterministic transitions between each set of states. Each robot has five possible actions; stay static, move right, move down, move left or move up. Robots are divided into two distinct classes; namely Scouts and Service-bots, each with specific capabilities and reward structures. The Scout agents can move twice as fast in any direction and have a POI activation range twice that of the Service-bots. This added mo-

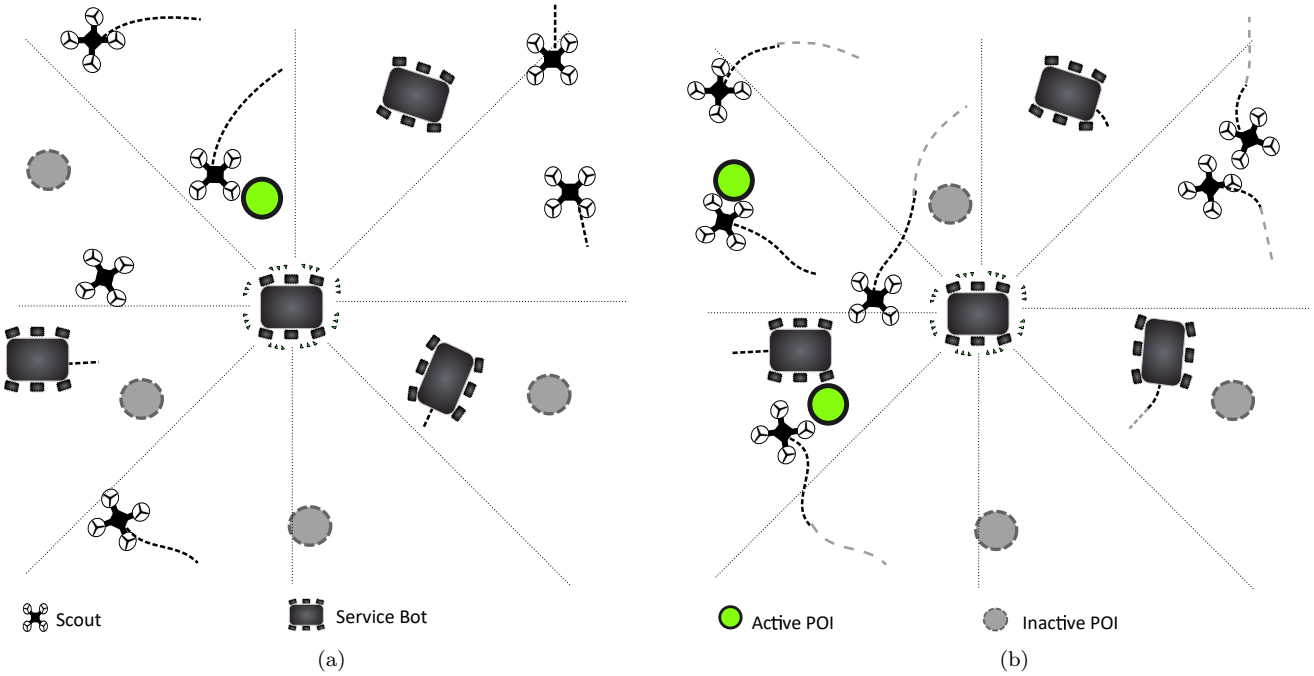


Figure 2: Illustration of our domain and sensor inputs discretized into sectors. Scout robots are shown here as UAVs and Service-bots as ground rovers. The world is divided up into 8 sectors. POIs are active for a limited time after it has been activated by a Scout robot. Inactive POIs are visible to Scout robots but Service-bots can only sense the ones activated at least once. (a) and (b) show the same domain within a couple of timesteps, and illustrate the dynamics of our environment where Scouts and Service-bots move around, and POIs become active and inactive as Scouts pass by

bility makes the Scouts more suited for exploration. In a real world scenario, the Scout agent can be thought of as a Unmanned Aerial vehicle (UAV) which covers a larger area faster and has a larger field of view. The Service-bots can then be thought of as ground units which are often slower and have limited visibility, often due to occlusion.

A robot from each class has to activate the POI within the same episode for the POI to be considered observed. This is similar to the heterogeneous problem formulation described in [10]. In this work however, the order in which these activations take place is also restricted such that a POI has to be activated by the Scout agent (scouted) first after which the Service-bot can access it, and only then is the POI considered to have satisfied its observation condition. This problem formulation implements a strict heterogeneous coupling requirement where the problem cannot be solved by a single robot from either class, and tight coordination, within, and between robots from each class is imperative.

3.2 Temporal Coupling Constraint

In addition to the tight heterogeneous coupling formulation, a key difficulty which is the primary point of investigation in this work are the constraints imposed by temporal coupling between the joint action required in strict order. Before a POI is activated by a Scout robot, it can be thought of as being in an inactive state. Once activated by a Scout, the POIs transition into an active state. A Service-bot may then access that POI to satisfy its observability requirement. In this paper, we introduce a temporal constraint ($t_{trigger}$) which limits the observability window during which the POI is active for. The Service agent has to access the POI within this window for the POI's observation requirement to be satisfied. Failing any such event within the accessibility window, the POI transitions back into an inactive state and the entire joint action has to be repeated. This constraint imposes a temporal coupling requirement orthogonal to the one posed before, and ensures that tight coordination between the heterogeneous robots in terms of both space and time is crucial to solve the task.

Temporal coupling constraints fall largely into two main categories: namely hard and soft coupling constraints. The soft coupling constraint allows for the Service-bot to access and successfully satisfy the POI's observation requirement at any timestep within $t_{trigger}$ following activation by a Scout robot. The hard coupling constraint further restricts Service-bot access and successful satisfaction of the POI's observation requirement to be at exactly $t_{trigger}$ timestep following activation by a Scout robot.

3.3 Robot capabilities

Each robot uses a neural network to map its sensor input to five output nodes which represent the probability distribution over its five possible actions. The robot's action is then simply the output node with the highest activation as shown in Equation 4.

$$action = arg\,max_a \{O(a)\} \tag{4}$$

For each robot, *action* represents the choice of action, a represents the set of all available actions and $O(a)$ represents the magnitude of activation at the output node corresponding to that particular action.

The input to the neural network for each robot is the

state of the world as perceived by the sensors corresponding to that robot. For each robot, the world is divided into eight separate sectors each spanning an angle of 45°. Each robot has six separate sensors that sense six separate measurements. The sensory data is then parsed and bundled for each sector to form the input for that particular sector. The full input state is then the concatenation of these measurements for each of the eight sectors as shown in Equation 5.

$$S_r = \sum_{i=1}^{k} \left\{ d^i_{POI}, n^i_{POI}, d^i_{Sc}, n^i_{Sc}, s^i_{Sb}, n^i_{Sb} \right\} \quad (5)$$

Here, S_r represents the perceived state for robot r which forms the input to its neural network, k represents the total number of sectors, d^i_{POI} and n^i_{POI} are the density and number of POIs in sector i respectively, d^i_{Sc} and n^i_{Sc} are the density and number of Scouts in sector i respectively, and finally, d^i_{Sb} and n^i_{Sb} are the density and number of Service-bots in sector i respectively.

The Scout and Service-bot sensors are identical apart from their capabilities for detecting POIs. The Scout robot can sense all POIs at any point in the episode while the Service-bot robot's POI sensors can only sense a POI once it has been activated, at least once by the Scout agents. This is identical to a scenario where the Service-bots don't actually have the ability to sense the POI but are dependent on the Scout robots to discover them and communicate their location. Once the Scout robots discover and activate a POI, this information is then broadcast to all the Service-bots which can now sense that POI.

3.4 Robot Objectives

In our experiments we compared two different evaluation functions to determine the performance of the robots: the system evaluation function and difference evaluation function [2]. Each version as implemented in our domain is described below:

- The **system evaluation** function captures the performance of the entire system (collection of robots across all classes) in one evaluation metric. This is the system reward which is the fundamental goal of learning. System evaluation gives this system reward metric obtained in an episode to all the robots in that group regardless of their own actions. This reward assignment technique guarantees that all robots are motivated by the same purpose.

- The **difference evaluation** function captures, for each robot, the impact that particular robot has on the full system performance [2]. For a robot r this is calculated by removing that robot, recalculating the system reward and computing the difference. Since the POIs that were successfully observed in our domain are relevant to calculating our system reward, the difference evaluation function for each robot r is locally computable.

4. CONTROL POLICY TRAINING

Each robot's control policy represents a mapping from its perceived state input to a probability distribution over its available actions. This mapping is approximated by a neural network which is evolved during training. MultiNEAT [4], a portable software library is used to implement NEAT in this work. Each robot from both robot classes have their own population of policies and a modified version of MultiNEAT, adjusted to handle a Cooperative Coevolutionary Algorithm (CCEA) with lenient learners is used to co-evolve them in parallel.

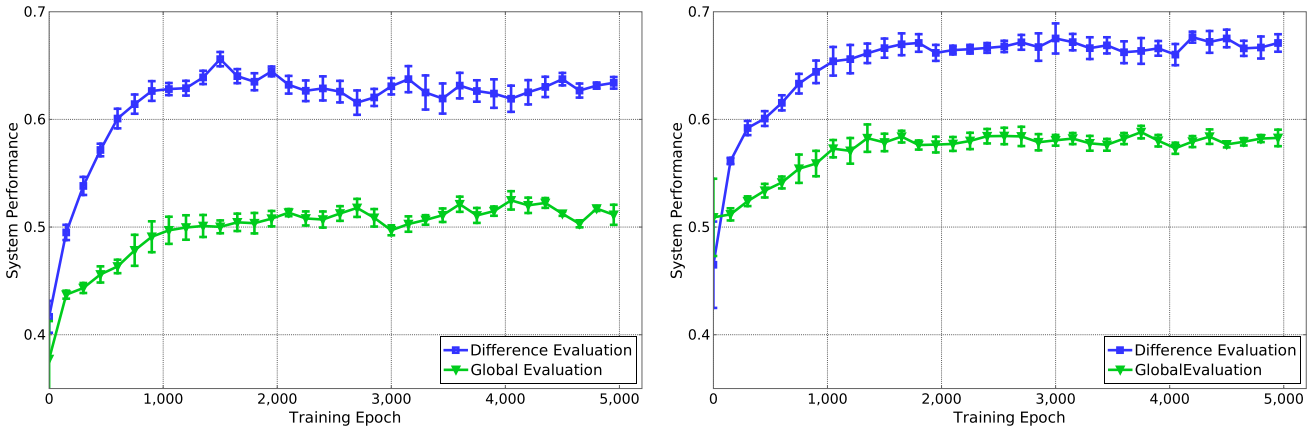
```
Initialize N populations of k genomes
foreach Epoch do
    foreach Generation do
        foreach Population do
            Produce k successor genomes using
            multipoint crossover
            Mutate successor genomes
            Build neural network from genome
        end
        foreach i = 2k * m do
            Select an agent from each population at
            random
            Add agents to team T_i
            Run simulation for team T_i
            Assign reward to members of T_i using the
            evaluation function selected
        end
        foreach Population do
            foreach Genome do
                fitness ← Maximum fitness achieved
            end
        end
        foreach Population do
            Select k best genomes
        end
    end
end
```

Algorithm 1: Lenient CCEA with MultiNEAT for temporally coupled heterogeneous multi-robot coordination

Algorithm 1 details the CCEA algorithm implemented on top of MultiNEAT with lenient learners. The number of populations N is equal to the total number of robots in our domain including both classes. The parameter k represents the size of each population which was set to 100 for these experiments. The parameter m is the number times each candidate was evaluated for each generation, set to 5 for our experiments. Each evaluation of the candidate was done in the context of different team members facilitated by the random team selection. This was done to reduce the effect of a candidate's teammates on the fitness of that candidate. Leniency [12] was implemented at the end of m evaluations (generations) where the maximum score achieved by a candidate is selected and assigned as fitness for that candidate.

5. RESULTS

The environment used for experimentation in this paper contained 4 Scout robots, 8 Service-bot robots and 12 POIs, striking a balance between amount of information gathered, and creating a congestion situation. The environment was highly dynamic where all the POIs spawned at different locations, determined randomly for each generation. This was implemented to necessitate coordination behavior based on sensory inputs rather than memorization of x-y coordinates.



(a) Stationary POIs

(b) Dynamic POIs

Figure 3: System performance of multiagent teams where a Service-bot have to access the POI within 3 timesteps of being activated by a Scout

The results are based on 5000 generations of learning. Averaged results of the system performance with error bars reporting the standard error in the mean are shown.

5.1 Soft Temporal Coupling

Figure 3 show the performance of multiagent coordination under the soft temporal constraint. Figure 3a shows the performance when the POIs are held stationary within the domain, once spawned at random locations, at the start of the episode. The heterogeneous teams trained with the difference evaluation perform significantly better than those trained with the global evaluation function, leading to approximately 18% improvement in system performance. This demonstrates the difference evaluation's capacity to capture the strictly ordered temporal coupling in our heterogeneous joint action space and form effective coordination policies that account for them. Difference evaluation effectively filters out the noise in the global evaluation signal and provides a clear learning signal for each agent that is commensurate to their marginal contribution. This robot-specific feedback is key to the process that allows our algorithm to be sensitive to the temporal constraint of our heterogeneous joint action space. Using this feedback tailored to each robot, NEAT is able to evolve robot policies that can account for the temporal coupling, and CCEA is able to parallelize the search process leading to effective coordination policies.

Figure 3b shows the performance when the POIs are no longer held stationary but instead move in a preordained path with a 25% probability for motion at each timestep. This setup effectively adds a layer of complexity to the coordination task, while also simultaneously adding opportunity. While the movement of the POIs imposes an added complexity for the heterogeneous team, the periodic nature of these paths add an opportunity for teams that are sensitive to it, and are able to exploit them. The result in figure 3b shows that our approach utilizing CCEA and NEAT is able to effectively handle this added complexity of periodically moving POIs and exploit it for increased system performance while using either difference or global evaluation, when compared to its stationary POI counterpart. Further, similar to the trend seen with stationary POIs, the heterogeneous teams trained with the difference evaluation perform significantly better than the one trained with global evaluation function leading to approximately 19% improvement in system performance. This demonstrates the robustness of our algorithm, utilizing agent-specific rewards facilitated by difference evaluation, when dealing with moving POIs.

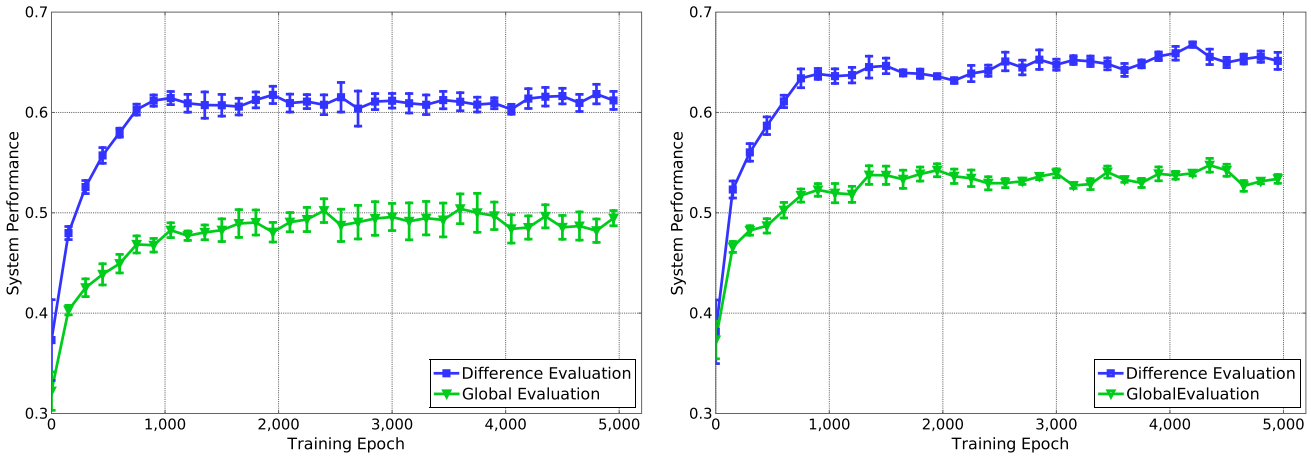
5.2 Hard Temporal Coupling

Figure 4 shows the performance of multiagent coordination under the hard temporal coupling setup. This imposes a tougher temporal constraint which restricts Service-bot access and successful satisfaction of the POI's observation requirement to be at exactly $t_{trigger}$ timesteps following activation by a Scout robot. Figure 4a shows the performance when the POIs are held stationary and figure 4b shows the performance when the POIs are dynamic. Similar to the trend seen for experiments with the soft coupling constraint, the heterogeneous teams trained with the difference evaluation perform significantly better than those trained with the global evaluation function in both experimental setups, leading to approximately 20% and 18% improvement in system performance, respectively. This result demonstrates the significant impact that the agent-specific reward assigned by difference evaluation has in capturing the strictly ordered temporal coupling in our heterogeneous joint action space, even when subjected to limiting constraints imposed by hard temporal coupling.

5.3 Temporal Constraint sensitivity

To probe our algorithm in terms of its ability to handle different margins of temporal constraints, we repeated the experiments described above with different temporal offsets (trigger time).

Figure 5 shows the results for temporal constraint sensitivity, for a hard temporal constraint setup with dynamic POIs. The heterogeneous teams trained with difference evaluation performed significantly better than those trained with global evaluation across all values of the temporal offsets. This monotonic gap highlights the robustness of our approach utilizing difference evaluation to successfully capture the strictly ordered temporal coupling in our heterogeneous joint action space. Interestingly, the performance gap between our approach using difference evaluation and the one using



(a) Stationary POIs

(b) Dynamic POIs

Figure 4: System performance of multiagent teams where a Service-bot have to access the POI exactly 3 timesteps after being activated by a Scout

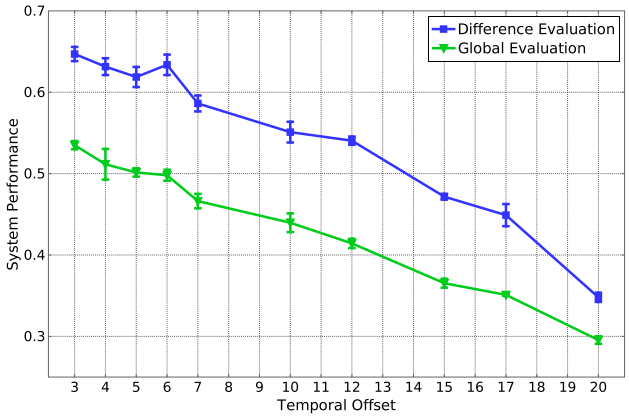


Figure 5: System performance when the Service-bot had to access the POI at varying time offsets of being activated by a Scout

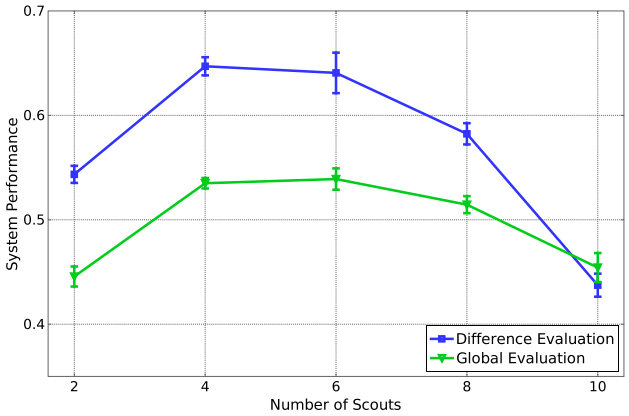


Figure 6: System performance across different team ratios under a hard coupling setup with dynamic POIs. The total number of robots in our heterogeneous team was held constant at 12. Number of Service-bot is 12 minus the number of Scouts

global evaluation diminishes with higher temporal offsets.

5.4 Team ratio sensitivity

To test the impact of team heterogeneity, we repeat the experiments described above with varying team ratios. The total number of robots in our experiments was held constant at 12, and the ratio of Scouts to Service-bots adding up to that total number was altered.

Figure 6 shows the results for team ratio sensitivity for a hard temporal constraint setup with dynamic POIs, and time offset (trigger time) set to 3. The heterogeneous teams trained with difference evaluation outperforms global evaluation in all but one team ratio composition, made up of 10 Scouts and 2 Service-bots. It's important to note that the Scout robots are twice as fast and have an influence region twice as large relative to the Service-bots. Thus, a team ratio skewed slightly towards having more number of Service-bots is ideal for overall system performance. This is reflected in the result as the 4-6 Scout range seems to perform the best for both approaches. A team composition of 10 Scouts and 2 Service-bots is very unbalanced as the 2 Service-bots are unable to cover a large part of the area and serve as huge bottlenecks. This greatly diminishes the value of agent specific rewards provided by difference evaluation.

5.5 Sample Trajectory

Figure 7 shows a series of snapshots for a coordination policy, operating in a randomly initialized environment. Each snapshot shows the position and trajectories of Scout robots, Service-bots and POIs. The coordination policies of the Scout robots and Service-bots were trained using our algorithm utilizing difference evaluation. After initialization, the Scout robots quickly disperse and activate the POIs while the slower Service-bots follow. Both classes of robots disperse purposefully, distributing themselves in proportion to the POIs in each direction, thus maximizing the potential for higher system performance. The Service-bots follow the Scout robots at close quarters, ensuring that the hard time

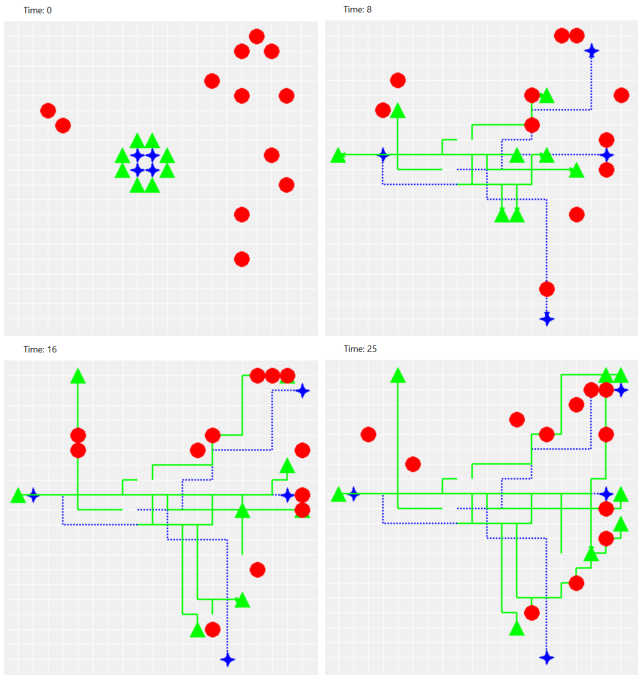


Figure 7: Sample path trajectory with 4 Scouts 8 Service-bots heterogeneous team when a Service-bot has to access a POI at exactly 3 timesteps following activation by a Scout robot. POIs are dynamic. Scouts are shown as blue crosshairs, Service-bots as green triangles and POIs as red circles. (top left) 0 timestep, (top right) time = 8, (bottom left) time = 16, (bottom right) time = 25

constraint of 3 timesteps is met, and succeed in satisfying those constraints most of the time. There are some sub-optimal behaviors seen as with the 1 Scout and 2 Service-bot mini-team operating on the left quarter of the domain. However, the strategy of approaching the two POIs and forming their own mini-team to pursue them, is a good stepping stone towards a better strategy. Overall, the heterogeneous team of robots exhibit good coordination and operate effectively in the given environment.

6. DISCUSSION

Real world coordination tasks often require a team of heterogeneous robots with varying capabilities to act in an ordered and time sensitive way. While the first requirement (heterogeneity) is a ripe area of research, the time coupling aspect which constrains the required joint action temporally, receives scant attention. In this paper we investigated this facet of task complexity by employing a heterogeneous multi-robot coordination domain, with tight temporal coupling requirements. We utilized a cooperative coevolutionary algorithm modified to evolve NEAT neural networks, in conjunction with difference evaluation, and successfully demonstrated that our approach is able to capture and account for the strictly ordered temporal coupling in heterogeneous joint action space, leading to effective coordination policies for these difficult tasks.

Further, the policies found through our approach were effective on both categories of temporal constraints namely, soft and hard, and robust to varying values of temporal constraint and team ratios. The work presented here focused its investigation on the problem formulation for an environmental monitoring task utilizing a proxy domain to enact its task requirements. The approach however is general and can be applied to various other real world multiagent coordination domains with similar requirements for heterogeneity and temporal constraints.

An example of such domain is underwater exploration, where a smaller, cheaper and more agile scout robot often explores the environment before a heavier, more expensive and more sophisticated robot travels to the location to analyze and collect samples. In this problem, it is vital that once a point of interest is scouted, the second robot analyze/collect samples within a specified time window, before the samples are swept away by water currents or fauna.

Further, an online trading setting could utilize two coordinating agents, the first of which analyzes the market, searches for investment opportunities and makes predictions while the second agent executes buy or sell actions. In volatile environments like this, a strict temporal coupling exists between the two heterogeneous agents' actions such that a correct joint action that maximizes system utility is also conditioned on meeting the strict temporal constraints. Our approach could be extended to form effective coordination policies in domains with these difficult requirements.

Future work will implement the coordination policies found using our approach in a physical robot. Work has begun in transcribing the individual robot policies found through our approach to be implemented on Pioneer Robots using the Robotic Operating System (ROS) environment, which would allows us to conduct real world experiments, and oversee real world testing and validation of the evolved policies in a physical environment.

REFERENCES

[1] T. Aaltonen, J. Adelman, T. Akimoto, B. A. Gonzalez, S. Amerio, D. Amidei, A. Anastassov, A. Annovi, J. Antos, G. Apollinari, et al. Observation of electroweak single top-quark production. *Physical review letters*, 103(9):092002, 2009.

[2] A. Agogino and K. Tumer. Efficient evaluation functions for evolving coordination. *Evolutionary Computation*, 16(2):257–288, 2008.

[3] M. Bowling and M. Veloso. Simultaneous adversarial multi-robot learning. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 699–704. Morgan Kaufmann Publishers Inc., 2003.

[4] P. Chervenski and S. Ryan. Multineat open source library. `http://multineat.com/`, 2012.

[5] M. Colby, J. J. Chung, and K. Tumer. Implicit adaptive multi-robot coordination in dynamic environments. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5168–5173. IEEE, 2015.

[6] M. Colby, L. Yliniemi, and K. Tumer. Autonomous multiagent space exploration with high-level human feedback. *Journal of Aerospace Information Systems*, pages 1–15, 2016.

[7] S. Devlin and D. Kudenko. Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and*

Multiagent Systems-Volume 1, pages 433–440, 2012.

[8] S. G. Ficici, O. Melnik, and J. B. Pollack. A game-theoretic and dynamical-systems analysis of selection methods in coevolution. *IEEE Transactions on Evolutionary Computation*, 9(6):580–602, 2005.

[9] Y. Ishiwaka, T. Sato, and Y. Kakazu. An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning. *Robotics and Autonomous Systems*, 43(4):245–256, 2003.

[10] M. Knudson and K. Tumer. Coevolution of heterogeneous multi-robot teams. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 127–134. ACM, 2010.

[11] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

[12] L. Panait, K. Tuyls, and S. Luke. Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *Journal of Machine Learning Research*, 9(Mar):423–457, 2008.

[13] E. Ruppin. Evolutionary autonomous agents: A neuroscience perspective. *Nature Reviews Neuroscience*, 3(2):132–141, 2002.

[14] K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time neuroevolution in the nero video game. *IEEE transactions on evolutionary computation*, 9(6):653–668, 2005.

[15] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[16] P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[17] K. Tuyls, P. J. Hoen, and B. Vanschoenwinkel. An evolutionary dynamical analysis of multi-agent learning in iterated games. *Autonomous Agents and Multi-Agent Systems*, 12(1):115–153, 2006.

[18] S. Whiteson and P. Stone. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, 7(May):877–917, 2006.

[19] C. H. Yong and R. Miikkulainen. Cooperative coevolution of multi-agent systems. *University of Texas at Austin, Austin, TX*, 2001.